SALUS SECURITY

SEP 2024

CODE SECURITY ASSESSMENT

ASTHERUS

Overview

Project Summary

- Name: AstherusVault
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/leek-z/astherus-contract
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	AstherusVault
Version	v2
Type	Solidity
Dates	Sep 13 2024
Logs	May 30 2024, Sep 13 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	4
Total informational issues	0
Total	5

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk	Medium	Centralization	Mitigated
2	Not applicable for rebase token	Low	Business Logic	Mitigated
3	Functions may be blocked if ERC20 token is paused	Low	Business Logic	Mitigated
4	Chainlink's latestRoundData() might return stale results	Low	Data Validation	Mitigated
5	Potential cross-contract/cross-chain signature replay	Low	Cryptography	Mitigated

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Centralization risk

Severity: Medium	Category: Centralization
Target: - contracts/AstherusVault.sol	

Description

In the AstherusVault contract, there are some privileged roles and accounts. The ADMIN_ROLE can set key configuration parameters and pause/unpause withdrawals.

If an account's private key is compromised, which has ADMIN_ROLE, an attacker can:

- Change the signer
- Withdraw any token in the contract

If the privileged accounts are plain EOA accounts, this can be worrisome and pose a risk to the other users.

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team. The team stated they will transfer to multi-sig accounts and implement timelock governors to enhance security.

2. Not applicable for rebase token

Severity: Medium	Category: Business Logic
Target: - contracts/AstherusVault.sol	

Description

When users deposit tokens, the contract will calculate the final token amount it receives and send the related event. Users can withdraw tokens with one valid signature.

The vulnerability is that the rebase token's balance in this contract can change even if there is no deposit()/withdraw(). This may lead to withdrawal reverted if there is not enough token balance in the contract.

contracts/AstherusVault.sol:L132-L141

```
function _transfer(address payable to, bool isNative, address currency, uint256 amount)
private {
    if (amount == 0) revert ZeroAmount();
    if (isNative) {
        to.sendValue(amount);
    } else {
        IERC20 token = IERC20(currency);
        require(token.balanceOf(address(this)) >= amount, "not enough currency balance");
        token.safeTransfer(to, amount);
    }
}
```

Recommendation

Consider not supporting rebase tokens.

Status

This issue has been acknowledged by the team. The team stated that we do not support rebase tokens.

3. Functions may be blocked if ERC20 token is paused

Severity: Low	Category: Business Logic
Target: - contracts/AstherusVault.sol	

Description

When users deposit tokens, the contract will calculate the final token amount it receives and send the related event. Users can withdraw tokens with one valid signature.

The vulnerability is that some ERC20 tokens support Pause functionality. If the ERC20 token is paused, the customers cannot withdraw tokens. This will block the whole function.

contracts/AstherusVault.sol:L132-L141

```
function _transfer(address payable to, bool isNative, address currency, uint256 amount)
private {
    if (amount == 0) revert ZeroAmount();
    if (isNative) {
        to.sendValue(amount);
    } else {
        IERC20 token = IERC20(currency);
        require(token.balanceOf(address(this)) >= amount, "not enough currency balance");
        token.safeTransfer(to, amount);
    }
}
```

Recommendation

When integrating with some ERC20 tokens, double check whether ERC20 tokens support pause functionality.

Status

This issue has been acknowledged by the team. The team explained that it's a risk control function. When integrating the token, they double-check that the functionality is disabled.

4. Chainlink’s latestRoundData() might return stale results

Severity: Low	Category: Data Validation
Target: - contracts/AstherusVault.sol	

Description

The _amountUsd() function is used to calculate the amount of USD corresponding to the amount of tokens a user will withdraw. It will retrieve the recent price from Chainlink’s latestRoundData(). According to Chainlink’s documentation, the answer will be updated when the value deviates beyond a specified threshold or when the heartbeat idle time has passed. However, if answers are not updated in time, the amount of USD will be calculated with a stale price, which may lead to unexpected withdrawal pause.

contracts/AstherusVault.sol:L188-L197

```
function _amountUsd(address currency, uint256 amount) private view returns (uint256) {
        Token memory token = supportToken[currency];
        uint256 price = token.price;
        if (!token.fixedPrice) {
        AggregatorV3Interface oracle = AggregatorV3Interface(token.priceFeed);
        (, int256 price_,,,) = oracle.latestRoundData();
        price = uint256(price_);
    }
    return price * amount * (10 ** USD_DECIMALS) / (10 ** (token.priceDecimals +
token.currencyDecimals));
}
```

Recommendation

It is recommended to track the updatedAt value from the latestRoundData() function to make sure that the answer is recent enough. If the reported answer is not updated within the heartbeat or within the acceptable time limits, consider rejecting it.

Status

This issue has been acknowledged by the team. The teams stated that Chainlink Price is not part of the core business and is only for user withdraw limit.

5. Potential cross-contract/cross-chain signature replay

Severity: Low	Category: Cryptography
Target: - contracts/AstherusVault.sol	

Description

Before withdrawing, the contract will verify if the signer has approved the message. However, the message does not contain the address of the contract and chain-id, making it vulnerable to cross-contract and cross-chain signature replay attacks.

contracts/AstherusVault.sol:L160-L163

```
function withdraw(bytes calldata message, bytes calldata signature) external
whenNotPaused {

require(signer.isValidSignatureNow(MessageHashUtils.toEthSignedMessageHash(keccak256(mes
sage)), signature), "only accept truthHolder signed message");
        (uint256 id, address payable to, bool isNative, address currency, uint256 amount,
uint256 deadline) =
                        abi.decode(message, (uint256, address, bool, address, uint256,
uint256));
```

Recommendation

Consider including the AstherusVault contract address and chain-id in the message to avoid cross-contract and cross-chain signature replay attacks.

Status

This issue has been acknowledged by the team. The teams stated that the truthholder is different between different chains, so there is no cross-chain risk.

2.3 Informational Findings

No informational issues were found.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 98606bc:

File	SHA-1 hash
contracts/AstherusVault.sol	e17c45e7e335d682d577cedbb43c9ed35053250d